Ranking

Jordan Boyd-Graber
University of Colorado Boulder
3. DECEMBER 2014
Roadmap

- Combining rankings: taking advantage of multiple weak rankers
- Maximum margin ranking: support vector machines
- Reduction to classification: optimizing
Roadmap

- Combining rankings: taking advantage of multiple weak rankers
- Maximum margin ranking: support vector machines
- Reduction to classification: optimizing
- Perhaps useful for project, if you’re creating new rankings
Ranking

- Web search (Google uses > 200 features)
- Movie rankings
- Dating
Plan

RankBoost

Maximum Margin Ranking

Classification and Other Objectives
An Efficient Boosting Algorithm for Combining Preferences

- **Feedback function**: \( \Phi : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \)
  - \( \phi(x_0, x_1) > 0 \): \( x_1 \) is preferred to \( x_0 \)
  - \( \phi(x_0, x_1) < 0 \): \( x_0 \) is preferred to \( x_1 \)
  - \( \phi(x_0, x_1) = 0 \): no preference

- **Want to learn distribution** \( D(x_0, x_1) \equiv c \cdot \max\{0, \phi(x_0, x_1)\} \) s.t.
  \[
  \sum_{x, x'} D(x, x') = 1 \quad (1)
  \]
What’s the goal?

- Minimize the number of misranked pairs under final ranking

\[
\sum_{x,x'} D(x, x') \cdot \mathbb{1}[H(x') \leq H(x)] = \Pr_{(x,y) \sim D}[H(y) \leq H(x)] \tag{2}
\]

- Choose entries with high weight in $D$ to be *important* (can’t get them wrong)
What’s the input

- Weak rankings of the form $h_t : \mathcal{X} \rightarrow \mathbb{R}$
- Could be different systems / users / feature sets
- Will combine them into a final ranking of the same form
What’s a weak ranking?

• A function of the form

\[ h(x) = \begin{cases} 
1 & \text{if } f_i(x) > \theta \\
0 & \text{if } f_i(x) \leq \theta \\
q_{\text{def}} & \text{if } f_i(x) == \bot 
\end{cases} \] (3)
What’s a weak ranking?

• A function of the form

\[
h(x) = \begin{cases} 
1 & \text{if } f_i(x) > \theta \\
0 & \text{if } f_i(x) \leq \theta \\
q_{\text{def}} & \text{if } f_i(x) = \bot 
\end{cases}
\]  

(3)

• How to find \( q_{\text{def}} \) and \( \theta \)?

• Binary search over how much it improves ranking implied by \( D \) (i.e., gets high weights right)
Algorithm

- Initialize $D_1$
- For $t = 1 \ldots T$:
  - Get weak ranking $h_t : \mathcal{X} \mapsto \mathbb{R}$
  - Choose $\alpha_t$
  - Update distribution

\[
D_{t+1}(x, y) \propto D_t(x, y) \exp \{ \alpha_t [h_t(x) - h_t(y)] \}
\]  

- Final ranking is

\[
H(x) = \sum_{1}^{T} \alpha_t h_t(x)
\]
Learning rate

- $\alpha_t$ encodes importance of individual weak learner
- In general decreases over iteration
- Find weighted discrepancy

$$r = \sum_{x,y} D(x, y) [h(y) - h(x)]$$

- Use $\alpha = \frac{1}{2} \ln \left( \frac{1+r}{1-r} \right)$
Learning rate

- \( \alpha_t \) encodes importance of individual weak learner
- In general decreases over iteration
- Find weighted discrepancy

\[
r = \sum_{x,y} D(x, y) [h(y) - h(x)]
\]  

\( (6) \)

- Use \( \alpha = \frac{1}{2} \ln \left( \frac{1+r}{1-r} \right) \)
- As \( r \) gets smaller, weak learner \( t \) will have lower weight
Performance

- Works better than individual features or nearest neighbor
Plan

RankBoost

Maximum Margin Ranking

Classification and Other Objectives
Examples as feature vectors

Every example has a feature vector $f(x)$
Turning features to rank

- Have a series of “levels” or ranks $y = 1 \ldots$
- We want to find a function to separate examples

$$f(x) \equiv \langle w \cdot \phi(x) \rangle$$ (7)
Maximizing the margin

\[ U(x) \]

\[ x_1 \]

\[ x_2 \]

\[ \theta(r_1) \]

\[ \theta(r_2) \]
Using SVM-light

- Each example has a rank
- and a query id
- and lots of features
Using SVM-light

# query 1
3 qid:1 1:1 2:1 3:0 4:0.2 5:0
2 qid:1 1:0 2:0 3:1 4:0.1 5:1
1 qid:1 1:0 2:1 3:0 4:0.4 5:0
1 qid:1 1:0 2:0 3:1 4:0.3 5:0

# query 2
1 qid:2 1:0 2:0 3:1 4:0.2 5:0
2 qid:2 1:1 2:0 3:1 4:0.4 5:0
1 qid:2 1:0 2:0 3:1 4:0.1 5:0
1 qid:2 1:0 2:0 3:1 4:0.2 5:0

# query 3
2 qid:3 1:0 2:0 3:1 4:0.1 5:1
3 qid:3 1:1 2:1 3:0 4:0.3 5:0
4 qid:3 1:1 2:0 3:0 4:0.4 5:1
1 qid:3 1:0 2:1 3:1 4:0.5 5:0
Plan

RankBoost

Maximum Margin Ranking

Classification and Other Objectives
Classification and Other Objectives

Are all pairs important?

- Often we care about the *top* of the result list
- Regression (as in previous section) not robust when there’s one right answer and many wrong ones
- Measured by the **AUC**: area under the curve
  - Imagine two classes: winners and losers
  - We want there to be a consecutive run of winners before losers in the results (extends to greater number of classes)
  - Want to minimize probability of losers before winners in an ordering $\pi$ on a set of examples $S = (x_1, y_1) \ldots$

$$I(\pi, S) = \frac{\sum_{i \neq j} [y_i > y_j] \pi(x_i, x_j)}{\sum_{i < j} [y_i \neq y_j]}$$  \hspace{1cm} (8)
Classification and Other Objectives

roc curve

AUC

True positive rate

False positive rate

0

0.2

0.4

0.6

0.8

1

0

0.2

0.4

0.6

0.8

1

0.2

0.4

0.6

0.8

1
Reduction to Classification

Robust Reductions from Ranking to Classification


- Produces a ranking using a classifier
- If regret of classifier is $r$, loss of classifier is at most $2r$
- Thus, if binary error rate is 20% due to inherent noise and 5% due to errors made by the classifier
- Then AUC regret is at most 10%
Algorithm

- **Learn a classifier**
  - Given a random pair of examples, learn a classifier \( c \) to predict whether it should prefer \( x_1 \) to \( x_2 \)
  - Return the classifier \( c \)

- **Get a ranking from the resulting classifier tournament**
  - For an example \( x \), define the degree
    \[
    \text{deg}(x) = |\{ x' : c(x, x') = 1, x' \in U \}|
    \]  
  - Sort by the degree of the node (number of matches it won)
Efficiency

- For ranking a large list, complexity $O(n^2)$ is unacceptable
- Possible to use variant of QuickSort $O(n \log n)$
- Has the same regret performance, but is randomized
Recap

- Ranking is an important problem
- Multiple approaches
  - Combining weak rankers
  - Max-margin
  - Tournament classification